Orderbook Architecture Document

1. Introduction

1.1. Purpose

This document provides a high level overview and explanation of an orderbook application. Contained within are sections that cover in detail the uses, implementation, and reasonings behind every component of the application.

1.2. Definitions, Acronyms, and Abbreviations

Term	Definition
Order Book	List of orders that show interest of buyers to sell or buy a financial instrument (stock)
Order	A purchase or sale inquiry for a stock
Buy Order	An inquiry to purchase a specific stock at a specific price
Sell Order	An inquiry to sell a specific stock at a specific price
Order ID	ID number for a given order
Size	The volume of stock for a given order
Side	Whether an order is a buy or sell order
Time (order)	The time at which an order is placed
Active	If the order listing is still available to execute
Offer Price	The sale or buy price of a share of a stock order
Symbol	Associated ticker symbol for a given stock
Transaction	The fulfillment of a given buy order and sell order
Buy Order ID	The Order ID associated with the buy order of a transaction

Sell Order ID	The Order ID associated with the sell order of a transaction
Final Time	The time at which a transaction is completed
Final Price	The price at which the transaction was executed
Amount	The volume of shares sold/bought in the transaction
Final Symbol	The ticker symbol for stock bought/sold in the transaction
Match	When the buy order price >= sell order price
Delete Order	Remove a buy or sell order from the orderbook
Trade History	List of all transaction that have occurred in the past

2. Architectural Goals and Constraints

2.1. Setup

The application is currently created to be hosted locally on a single machine. While running it will send and retrieve information from a MySQL database stored on the same machine (Database design is covered in a later section). This program can be adjusted to work with databases on different machines, however this was not the main focus for this version of the application.

2.2. Reliability

The application has been vigorously tested to ensure that all logic is sound and any errors that may arise have been accounted for and will be dealt with by the application should they arise.

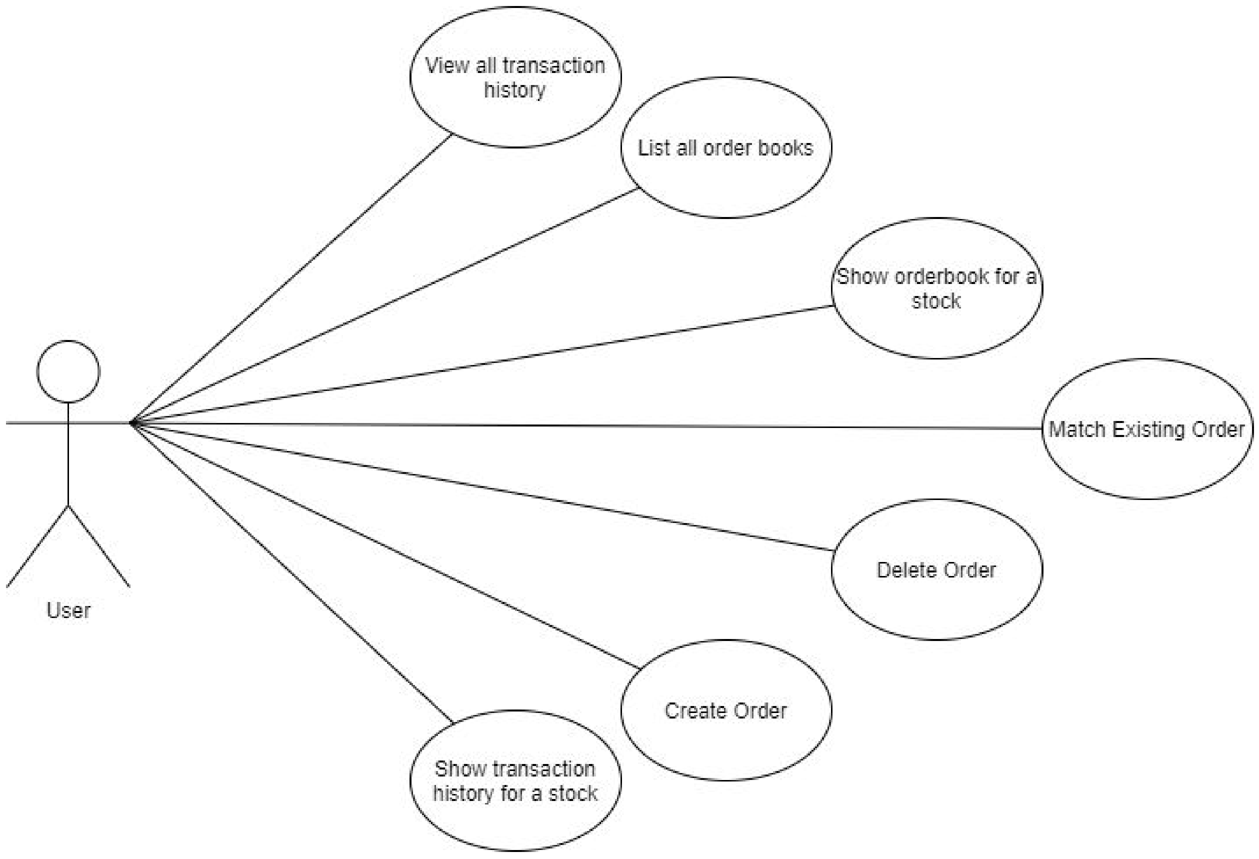
2.3. Development tools

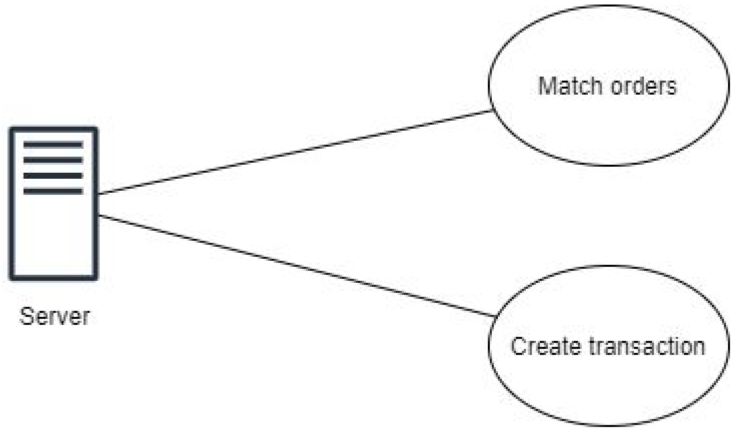
This application was created using the following tools:

Database: MySQL
Diagrams: Draw.io
Programming: NetBeans IDE, Visual Studio Code

3. Use-Case View

3.1. Use case diagrams


View all transaction history
List all order books
Show orderbook for a stock
Match Existing Order
Delete Order
Create Order
Show transaction history for a stock
User



3.2. Use case realizations

3.2.1. User specific use cases

3.2.1.1. View transaction history for symbol from navigation bar

Use case name	View transaction history for symbol from navigation bar
Scenario	User wants to view all transaction history for an orderbook for a specific symbol using the navigation bar.
Triggering Event	Clicking symbol name from “View Trade History” dropdown menu in navigation bar.
Brief Description	On click, a drop down menu will appear listing all symbols currently tracked by the application. From this list, a user needs to choose a symbol to be redirected to an orderbook for said symbol.
Actors	User
Related use cases	View transaction history for symbol from home page
Preconditions	None
Post conditions	None
Flow of events	User loads any webpage for the program. User chooses “View History” from the navigation bar. User chooses a symbol from the drop down list. Application requests all transactions from the connected database. Application adds transactions to the webpage via model. Application redirects users to the webpage. Webpage populates with the given model.
Exception conditions	If the symbol cannot be found, the user will be redirected to an error page asking them if they would like to create an orderbook for said symbol.

3.2.1.2. View transaction history for symbol from home page

Use case name	View transaction history for symbol from home page
Scenario	User wants to view all transaction history for an orderbook for a specific symbol from the home page
Triggering Event	Clicking “View Trade History” button in symbol activity form
Brief Description	On click, the user will be redirected to the trade history for a selected symbol’s orderbook.
Actors	User
Related use cases	View transaction history for symbol from navigation bar
Preconditions	None
Post conditions	None
Flow of events	User loads the home page User chooses a symbol from the drop down menu in “Symbol Activity”. User chooses “View Trade History” directly below the symbol drop down menu. Application requests all transactions from the connected database. Application adds transactions to the webpage via model. Application redirects users to the webpage. Webpage populates with the given model.
Exception conditions	If the symbol cannot be found, the user will be redirected to an error page asking them if they would like to create an orderbook for said symbol.

3.2.1.3. Show orderbook for a stock symbol from navigation bar

Use case name	Show orderbook for a stock symbol from navigation bar
Scenario	User wants the orderbook for a specific stock using the navigation bar

Triggering Event	User chooses a symbol from the “View an Orderbook” drop down menu in the navigation bar.
Brief Description	Clicking on the button “View Orderbook” will open a drop down list containing all symbols currently tracked in the application.
Actors	User
Related use cases	Show orderbook for a stock symbol from home page
Preconditions	None
Post conditions	None
Flow of events	User clicks “View an Orderbook” from the navigation bar. User chooses a symbol from the opened drop down menu. Application gets all orders for said symbol and adds them to the model. Application redirects user to a webpage to display orders. Webpage populates with all orders from the model.
Exception conditions	None.

3.2.1.4. Show orderbook for a stock symbol from home page

Use case name	Show orderbook for a stock symbol from navigation bar
Scenario	User wants the orderbook for a specific stock using the navigation bar
Triggering Event	User clicks “View Orderbook” in symbol activity form
Brief Description	On click, the user will be redirected to the orderbook for the selected symbol.
Actors	User
Related use cases	Show orderbook for a stock symbol from navigation bar
Preconditions	None
Post conditions	None

Flow of events	User chooses the symbol from the drop down menu located in the symbol activity form. User chooses the “View Orderbook” button directly below the chosen symbol. Application gets all orders for said symbol and adds them to the model. Application redirects user to a webpage to display orders. Webpage populates with all orders from the model.
Exception conditions	None.

3.2.1.5. Match existing order

Use case name	Match existing order
Scenario	User matches top buy/sell order
Triggering Event	User clicks “Match” button next to an order
Brief Description	On click, an order will be generated that will match the chosen order (price, amount, etc.) with the exception of the order type (this will be the opposite). Once created, a transaction is conducted using the two orders, fulfilling both and creating/updating the relevant entries in the database for storage.
Actors	User
Related use cases	Make transaction
Preconditions	At least one order must exist and the given order must be at the top of the list for their type (highest buy price or lowest sell price).
Post conditions	Orders should be fulfilled and a transaction connecting the two should be created. All three will be updated/saved to the database.
Flow of events	User chooses the top order they want to fulfill. An opposite order is generated based on the order chosen. Orders are used to create a transaction, which is stored in the database along with the updated(fulfilled) orders.

Exception conditions	If no orders exist, this operation is not available to the user.

3.2.1.6. Delete order

Use case name	Delete order
Scenario	User chooses to delete an order from an orderbook for a given symbol
Triggering Event	User chooses delete for an order
Brief Description	On click, the selected order will either be deleted completely from the database or will be considered inactive (size will be set to zero) so it will not be displayed. This is done only if the selected order has been previously used for a transaction(s).
Actors	User
Related use cases	None
Preconditions	A order must exist
Post conditions	An order considered inactive by the application (size will be 0) or the order will be deleted.
Flow of events	User clicks “Delete” for an order. Order is checked against the database to see if any transactions exist that reference said order. If such a transaction exists, the order’s size is set to zero to be considered inactive. Otherwise it will be deleted completely from the database.
Exception conditions	None

3.2.2. Automated use cases

3.2.2.1. Match orders

Use case name	Match Orders
Scenario	Two orders are compared to see if a transaction is possible.

Triggering Event	Triggers via a timer within the application.
Brief Description	For two given orders, the application will compare the elements of each order. This includes the price, the stock symbol and the amount. If a transaction can be created using the two orders, it will be created.
Actors	Application
Related use cases	Create transaction
Preconditions	Two orders capable of a transaction
Post conditions	Updated orders and a new transaction.
Flow of events	Two orders are collected for comparison. Comparisons are done on price, amount, and symbol. Assuming symbol matches, the amount can be at least partially fulfilled for one and completely for the other, the orders are sent to create a transaction.
Exception conditions	Orders are not compatible, in which case no transaction is made.

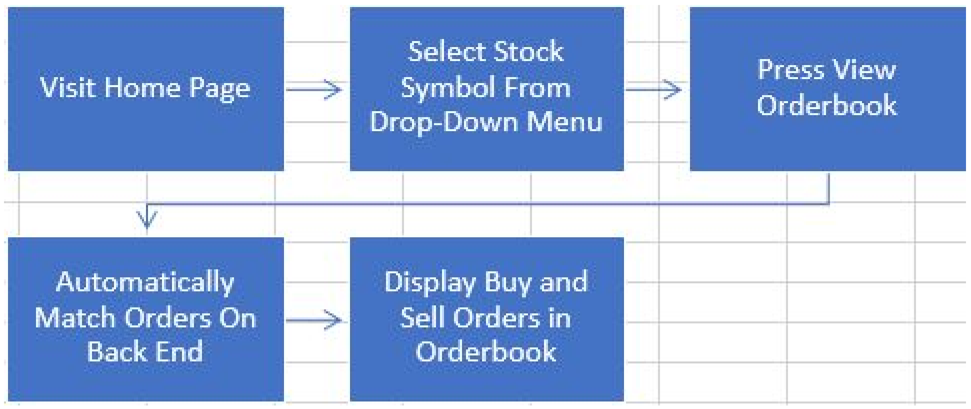
3.2.2.2. Create transaction

Use case name	Create transaction
Scenario	Two orders are used to create a transaction, updating the orders to show the proper values as well.
Triggering Event	Called by “Match Orders” and “Match Existing Order”. See related for more information.
Brief Description	Two orders are used to create a new transaction object. This transaction will contain the price, amount of stocks traded, symbol of the stocks, and date of creation. This will be recorded and the orders used will have their values in storage updated to reflect the change in amount.
Actors	Application
Related use cases	Match Orders, Match Existing Order

Preconditions	Two orders that are compatible.
Post conditions	Two updated orders and a newly created transaction.
Flow of events	Two orders are sent in. The necessary data for the transaction is pulled from the orders. The symbol will match the two orders. The price will be set to equal the buy order price. The amount will be set to the lowest amount value of either order. Orders will be updated to reflect their new amount values. Transaction will be recorded.
Exception conditions	None.

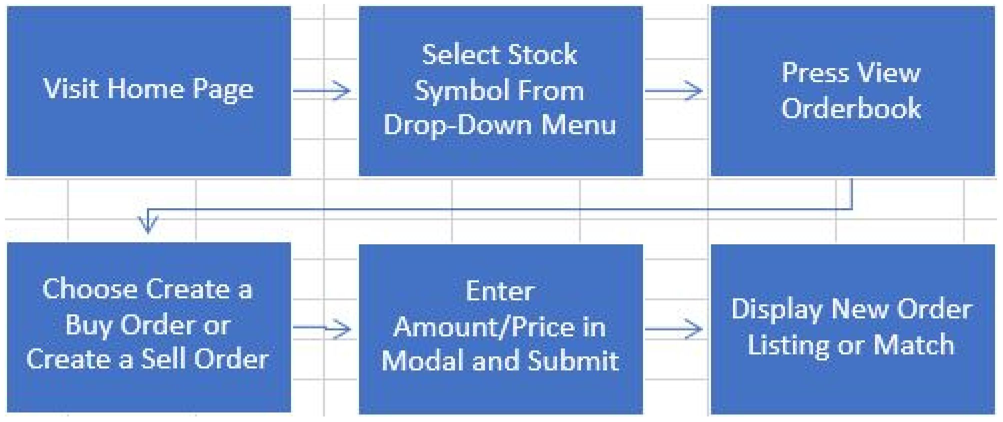
4. Activity Diagrams

4.1. Automatic Order Matching/Runtime processes

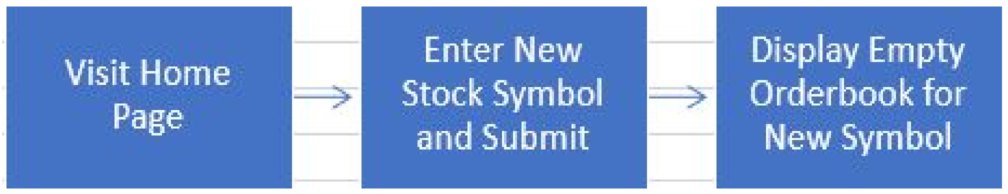


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Automatically Match Orders On Back End
Display Buy and Sell Orders in Orderbook

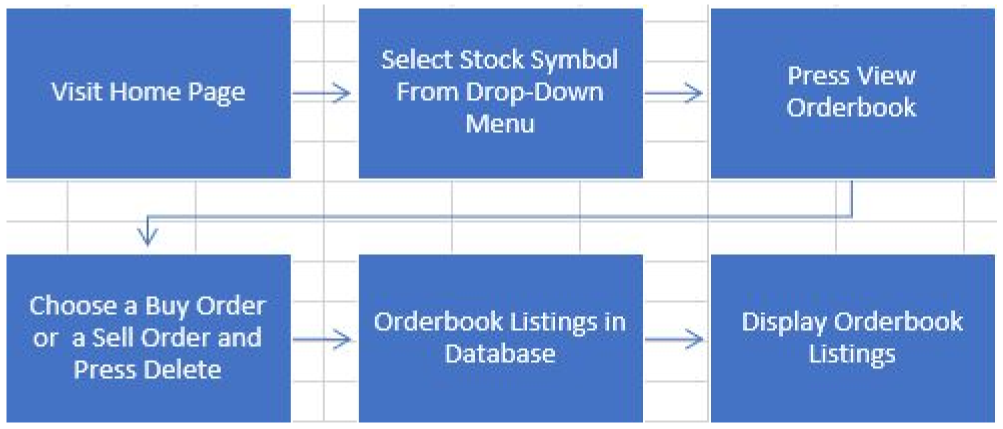
4.2. Adding an Order


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Choose Create a Buy Order or Create a Sell Order
Enter Amount/Price in Modal and Submit
Display New Order Listing or Match

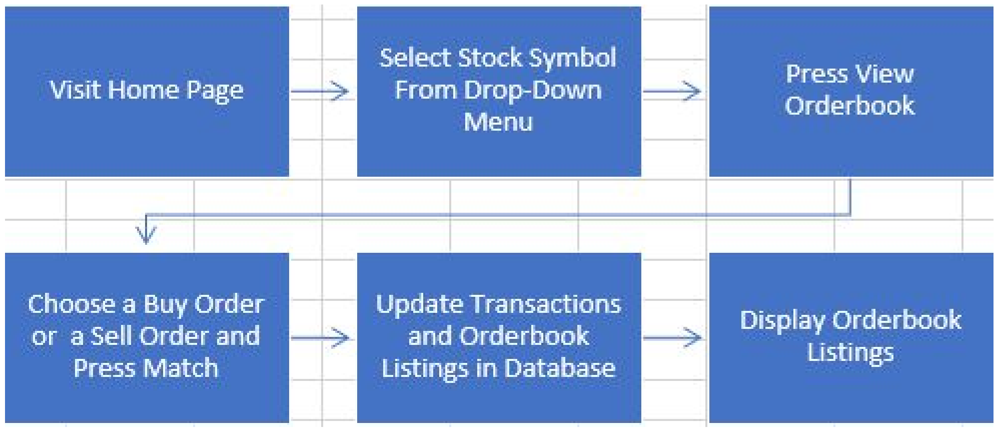
4.3 Create New Orderbook


Visit Home Page
Enter New Stock Symbol and Submit
Display Empty Orderbook for New Symbol

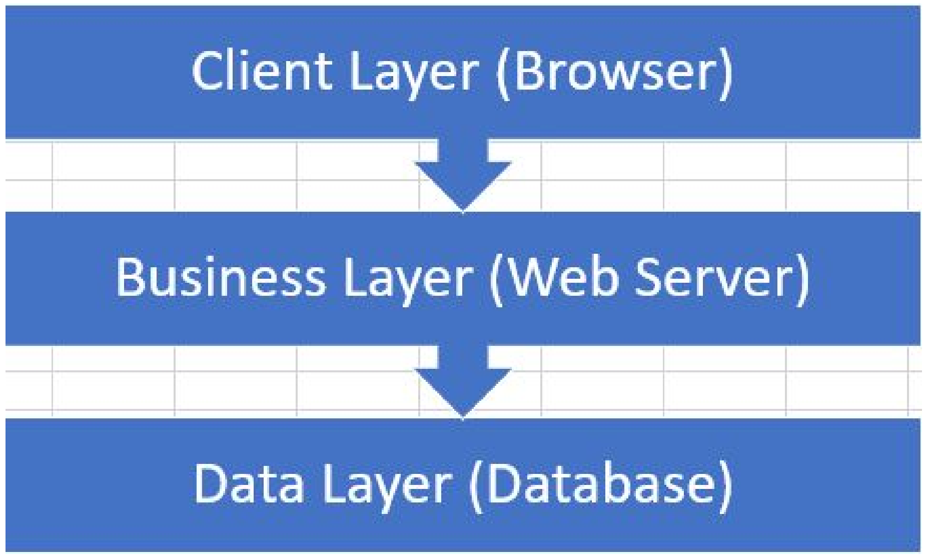
4.4 Deleting an Order


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Choose a Buy Order or a Sell Order and Press Delete
Orderbook Listings in Database
Display Orderbook Listings

4.5 Manual Order Matching


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Choose a Buy Order or a Sell Order and Press Match
Update Transactions and Orderbook Listings in Database
Display Orderbook Listings

5. Implementation View



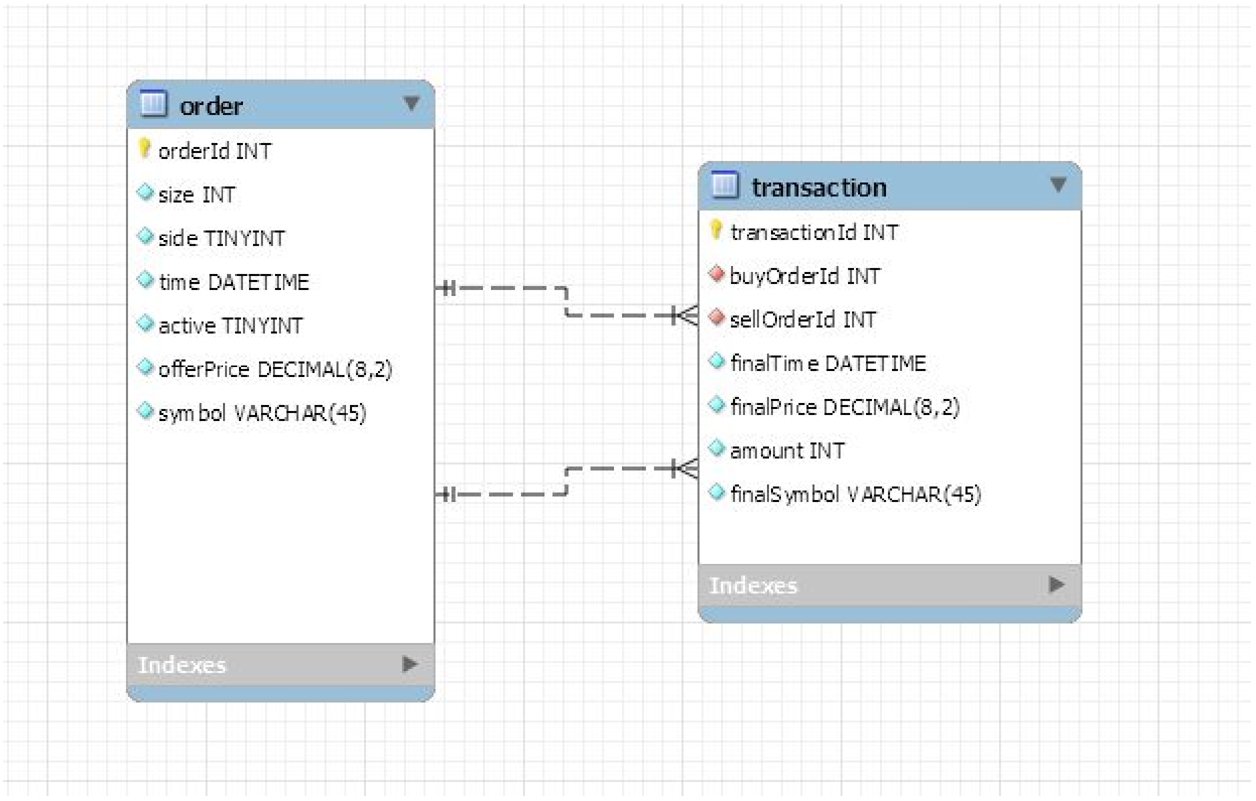
The application was created in the style of MVC. The reasoning of this was to control what each layer could access to avoid unnecessary/unexpected changes to the data and to other aspects of the application.

View layer (client layer) is the user interface (web pages). The actions the user takes in these web pages is handled by controller classes. These classes in turn call functions from the model classes to handle data and business logic. Doing this has the added benefit of reducing complexity by reducing the sizes of each function.

The model classes are separated into two layers, the business layer and data layer. The business layer contains all business logic and connects the view layer (controller) with the data layer. The data layer in turn connects the business layer with the connected database and is directly responsible for the saving and retrieving of data from a database.

6. Data View

6.1. ER diagram



6.2. MySQL implementation

The database was designed in MySQL. The main tables are:

1. Order
2. Transaction

Hibernate framework is used to automate the relations between the two tables. Classes were also created for each table in the database, making it possible for the application to work with the database's data.